

Generate images with SDXL Turbo

When it comes to working with an AI image generator, the speed in which images are generated is often a compromise. RunPod's Serverless Workers allows you to host [SDXL Turbo](#) from Stability AI, which is a fast text-to-image model.

In this tutorial, you'll build a web application, where you'll leverage RunPod's Serverless Worker and Endpoint to return an image from a text-based input.

By the end of this tutorial, you'll have an understanding of running a Serverless Worker on RunPod and sending requests to an Endpoint to receive a response.

You can proceed with the tutorial by following the build steps outlined here or skip directly to [Deploy a Serverless Endpoint](#) section.

Prerequisites

This section presumes you have an understanding of the terminal and can execute commands from your terminal.

Before starting this tutorial, you'll need access to:

RunPod

To continue with this quick start, you'll need access to the following from RunPod:

- RunPod account
- RunPod API Key

Docker

To build your Docker image, you'll need access to the following:

**Ask AI**

- Docker installed
- Docker account

You can also use the prebuilt image from [runpod/sdxl-turbo](#).

GitHub

To clone the `worker-sdxl-turbo` repo, you'll need access to the following:

- Git installed
- Permissions to clone GitHub repos

With the prerequisites covered, get started by building and pushing a Docker image to a container registry.

Build and push your Docker image

This step will walk you through building and pushing your Docker image to your container registry. This is useful to building custom images for your use case. If you prefer, you can use the prebuilt image from [runpod/sdxl-turbo](#) instead of building your own.

Building a Docker image allows you to specify the container when creating a Worker. The Docker image includes the [RunPod Handler](#) which is how you provide instructions to Worker to perform some task. In this example, the Handler is responsible for taking a Job and returning a base 64 instance of the image.

1. Clone the [RunPod Worker SDXL Turbo](#) repository:

```
gh repo clone runpod-workers/worker-sdxl-turbo
```

2. Navigate to the root of the cloned repo:

```
cd worker-sdxl-turbo
```

3. Build the Docker image:

```
docker build --tag <username>/<repo>:<tag> .
```

4. Push your container registry:



Ask AI

```
docker push <username>/<repo>:<tag>
```

Now that you've pushed your container registry, you're ready to deploy your Serverless Endpoint to RunPod.

Deploy a Serverless Endpoint

The container you just built will run on the Worker you're creating. Here, you will configure and deploy the Endpoint. This will include the GPU and the storage needed for your Worker.

This step will walk you through deploying a Serverless Endpoint to RunPod.

1. Login to the [RunPod Serverless console](#).
2. Select **+ New Endpoint**.
3. Provide the following:
 - i. Endpoint name.
 - ii. Select a GPU.
 - iii. Configure the number of workers.
 - iv. (optional) Select **FlashBoot**.
 - v. (optional) Select a template.
 - vi. Enter the name of your Docker image.
 - For example, `runpod/sdx1-turbo:latest`.
 - vii. Specify enough memory for your Docker image.
4. Select **Deploy**.

Now, let's send a request to your Endpoint.

Send a request

Now that our Endpoint is deployed, you can begin interacting with and integrating it into an application. Before writing the logic into the applicaiton, ensure that you can interact with the Endpoint by sending a request.

Run the following command:

cURL

Output



Ask AI

```
curl -X POST "https://api.runpod.ai/v2/${YOUR_ENDPOINT}/runsync" \
-H "accept: application/json" \
-H "content-type: application/json" \
-H "authorization: ${YOUR_API_KEY}" \
-d '{
  "input": {
    "prompt": "${YOUR_PROMPT}",
    "num_inference_steps": 25,
    "refiner_inference_steps": 50,
    "width": 1024,
    "height": 1024,
    "guidance_scale": 7.5,
    "strength": 0.3,
    "seed": null,
    "num_images": 1
  }
}'
```

Export your variable names in your terminal session or replace them in line:

- `YOUR_ENDPOINT`: The name of your Endpoint.
- `YOUR_API_KEY`: The API Key required with read and write access.
- `YOUR_PROMPT`: The custom prompt passed to the model.

You should see the output. The status will return `PENDING`; but quickly change to `COMPLETED` if you query the Job Id.

Integrate into your application

Now, let's create a web application that can take advantage of writing a prompt and generate an image based on that prompt. While these steps are specific to JavaScript, you can make requests against your Endpoint in any language of your choice.

To do that, you'll create two files:

- `index.html`: The frontend to your web application.
- `script.js`: The backend which handles the logic behind getting the prompt and the call to the Serverless Endpoint.

HTML

JavaScript



Ask AI

The HTML file (`index.html`) sets up a user interface with an input box for the prompt and a button to trigger the image generation.

```
<!-- index.html -->
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>RunPod AI Image Generator</title>
  <style>
    body {
      font-family: Arial, sans-serif;
      text-align: center;
      padding: 20px;
    }

    #imageResult {
      margin-top: 20px;
    }
  </style>
</head>
<body>
  <h1>RunPod AI Image Generator</h1>
  <input type="text" id="promptInput" placeholder="Enter your image prompt">
  <button onclick="generateImage()">Generate Image</button>

  <div id="imageResult"></div>

  <script src="script.js"></script>
</body>
</html>
```

1. Replace `${process.env.REACT_APP_AUTH_TOKEN}` with your actual API key.
2. Replace `${process.env.REACT_APP_ENDPOINT_ID}` with your specific Endpoint.
3. Open `index.html` in a web browser, enter a prompt, and select **Generate Image** to see the result.

This web application serves as a basic example of how to interact with your RunPod serverless endpoint from a client-side application. It can be expanded or modified to fit more complex use cases.



Ask AI

Run a server


You can run a server through Python or by opening the `index.html` page in your browser.

Python

File explorer

Run the following command to start a server locally using Python.

```
python -m http.server 8000
```

 [Edit this page](#)

Previous

« [Run your first serverless endpoint with Stable Diffusion](#)

Next

[Run your first AI API with Stable Diffusion](#) »

Docs


[Overview](#)

[Tutorials](#)

[AI APIs](#)

Community

[Discord](#) 

[Contact us](#) 

More

[Blog](#) 

[GitHub](#) 



Ask AI

